Introduction to Greedy Approach, Greedy Coin Change, Greedy Bin Packing

Week-01, Lecture-02

Course Code: CSE221

Course Title: Algorithms

Program: B.Sc. in CSE

Course Teacher: Tanzina Afroz Rimi

Designation: Lecturer

Email: tanzinaafroz.cse@diu.edu.bd

Greedy Algorithm

- Greedy algorithms make the choice that looks best at the moment.
- This locally optimal choice may lead to a globally optimal solution (i.e. an optimal solution to the entire problem).

When can we use Greedy algorithms?

We can use a greedy algorithm when the following are true:

1) **The greedy choice property:** A globally optimal solution can be arrived at by making a locally optimal (greedy) choice.

2) **The optimal substructure property:** The optimal solution contains within its optimal solutions to subproblems.

Designing Greedy Algorithms

1. Cast the optimization problem as one for which:
   - we make a choice and are left with only one subproblem to solve
2. Prove the GREEDY CHOICE
   - that there is always an optimal solution to the original problem that makes the greedy choice
3. Prove the OPTIMAL SUBSTRUCTURE:
   - the greedy choice + an optimal solution to the resulting subproblem leads to an optimal solution

Example: Making Change

- Instance: amount (in cents) to return to customer
- Problem: do this using fewest number of coins
- Example:
  - Assume that we have an unlimited number of coins of various denominations:
    - 1c (pennies), 5c (nickels), 10c (dimes), 25c (quarters), 1$ (loonies)
  - Objective: Pay out a given sum $5.64 with the smallest number of coins possible.

The Coin Changing Problem

- Assume that we have an unlimited number of coins of various denominations:
  - 1c (pennies), 5c (nickels), 10c (dimes), 25c (quarters), 1$ (loonies)
- Objective: Pay out a given sum S with the smallest number of coins possible.

- <u>The greedy coin changing algorithm</u>:
  - This is a $\Theta(m)$ algorithm where m = number of denominations.

```
while S > 0 do
    c := value of the largest coin no larger than S;
    num := S / c;
    pay out num coins of value c;
    S := S - num*c;
```

Example: Making Change

- E.g.:

  $5.64 = $2 +$2 + $1 +
  .25 + .25 + .10 +
  .01 + .01 + .01 +.01

Making Change – A big problem

- Example 2: Coins are valued $.30, $.20, $.05, $.01
  - Does not have greedy-choice property, since $.40 is best made with two $.20's, but the greedy solution will pick three coins (which ones?)

Bin Packing algorithm

Fitting things neatly & efficiently inside a larger container

6 groups of people, of group sizes 3, 1, 6, 4, 5 and 2 need to fit onto minibuses with capacity 7 but must stay together in their groups. Find the number of minibuses need to pack them in efficiently and so that each group stays together.

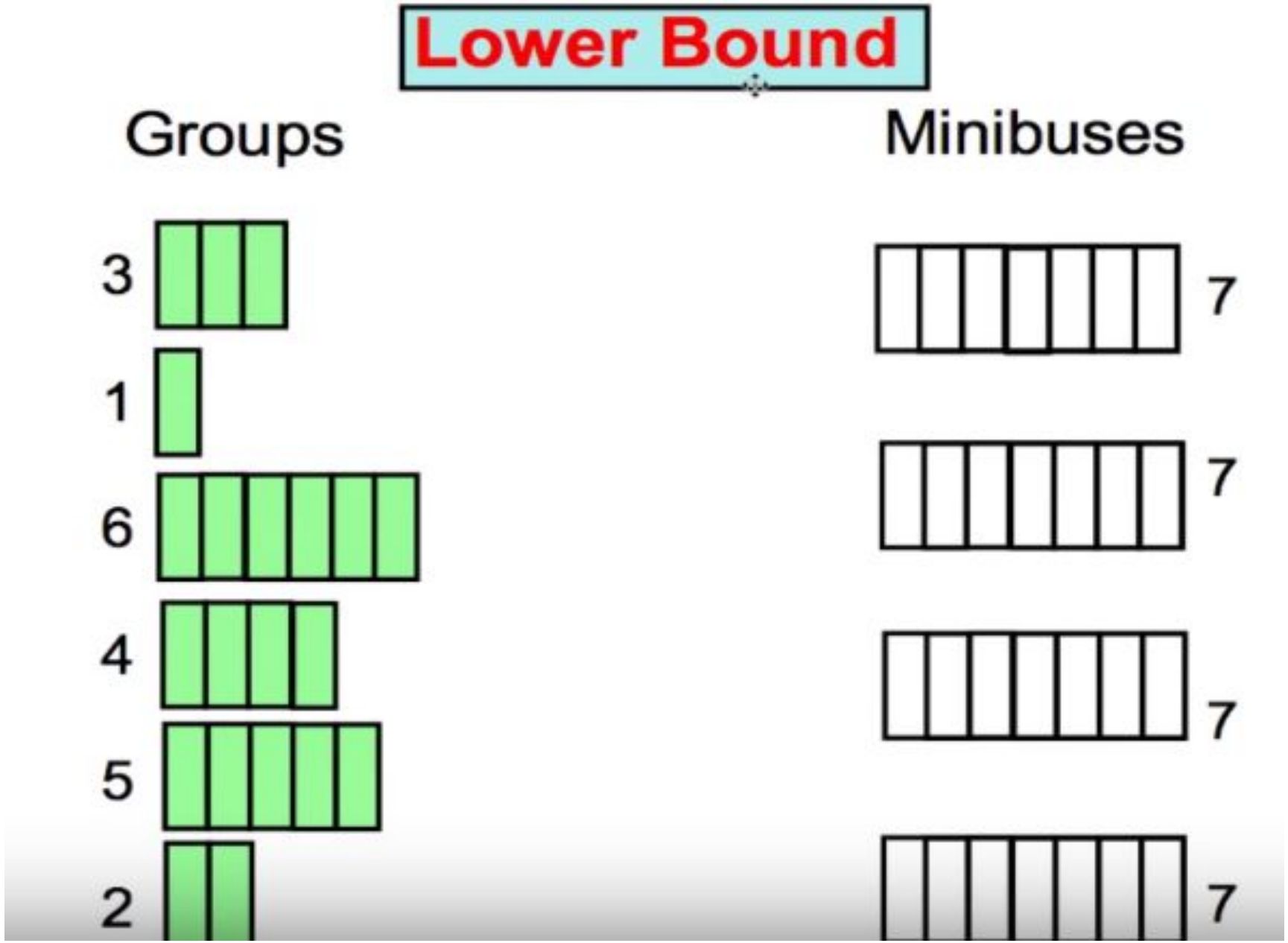
Bin Packing algorithms

Four things you needs to know

1. How to find the **lower bound** for the problem
2. How to perform the **first-fit** algorithm
3. How to perform the **first-fit decreasing** algorithm
4. How to perform **full-bin packing**

6 groups of people, of group sizes 3, 1, 6, 4, 5 and 2 need to fit onto minibuses with capacity 7 but must stay together in their groups.
Find the number of minibuses need to pack them in efficiently and so that each group stays together.

Lower Bound



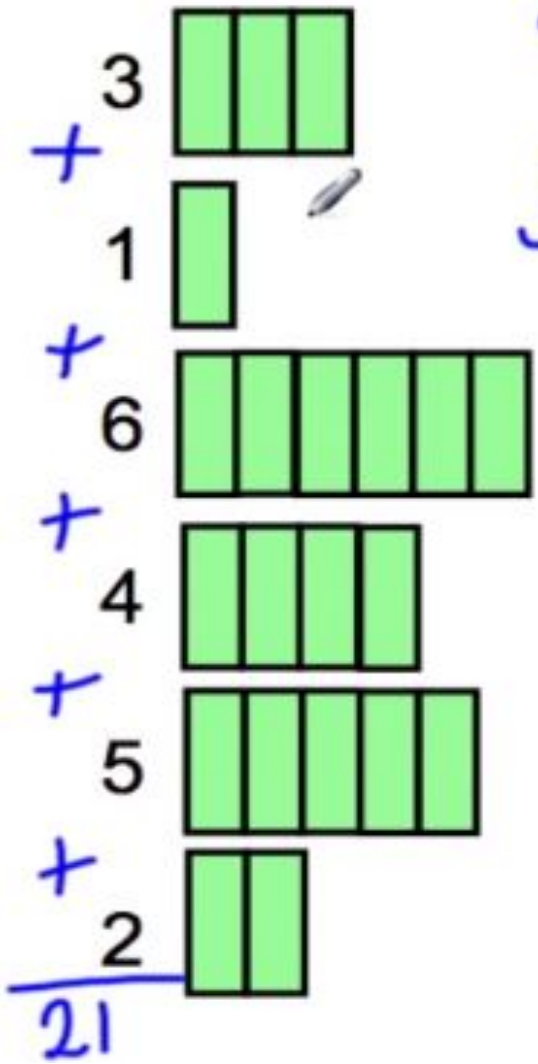
Lower Bound

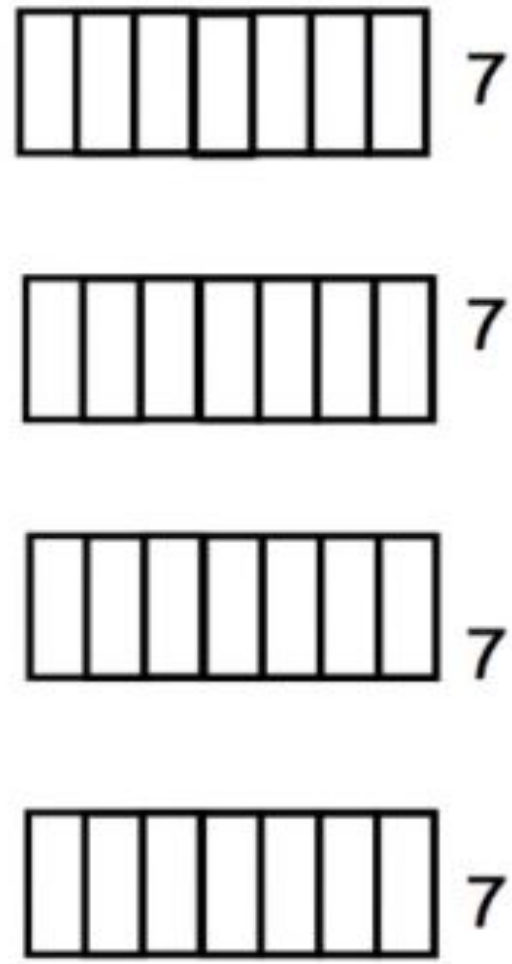
Groups

Minibuses

$21 \div 7 = 3$

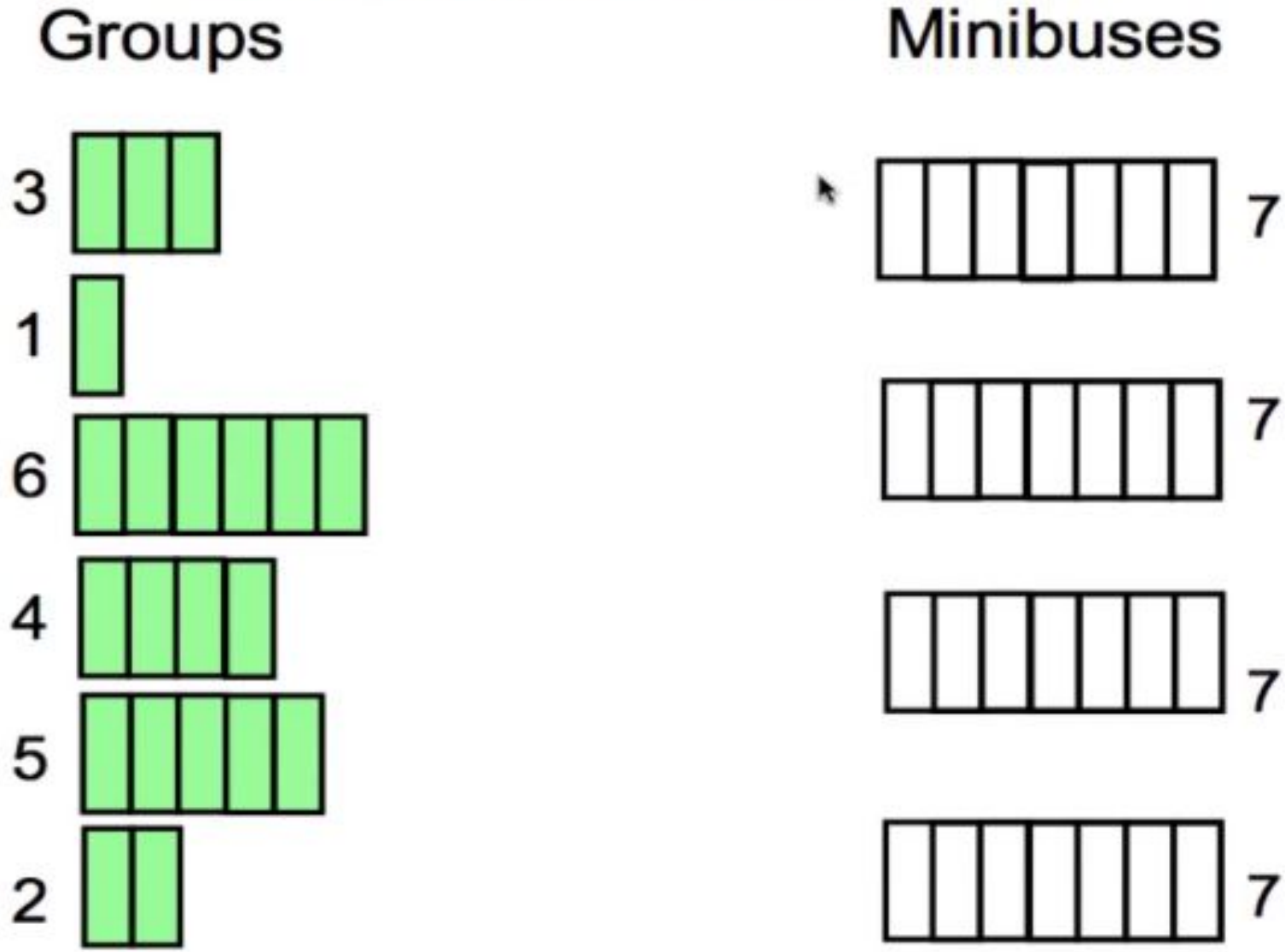
3 lower bound





Extend Page

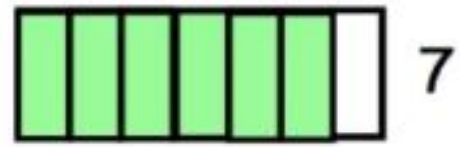
First-fit algorithm

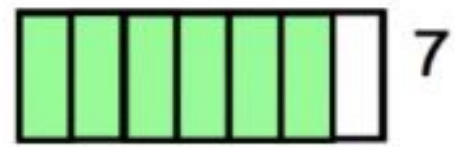


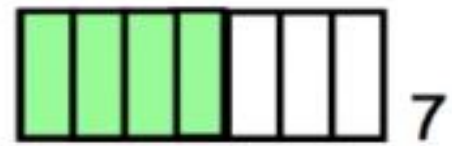
Extend Page

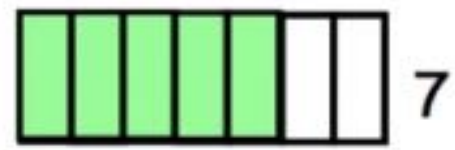
First-fit algorithm

Groups	Minibuses	
3	4 minibuses	7
1	7 spaces	7
6		
4		7
5		
2		7



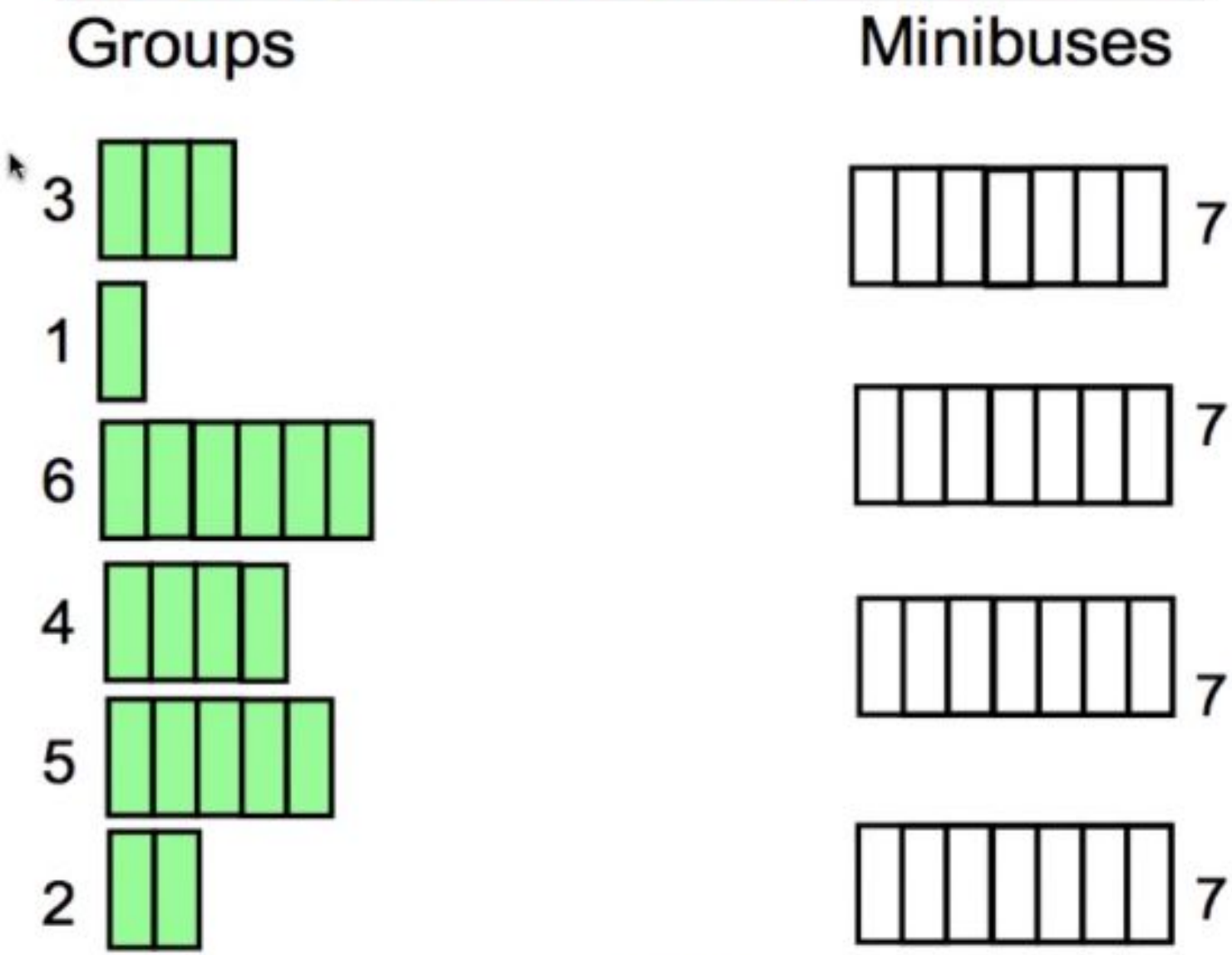






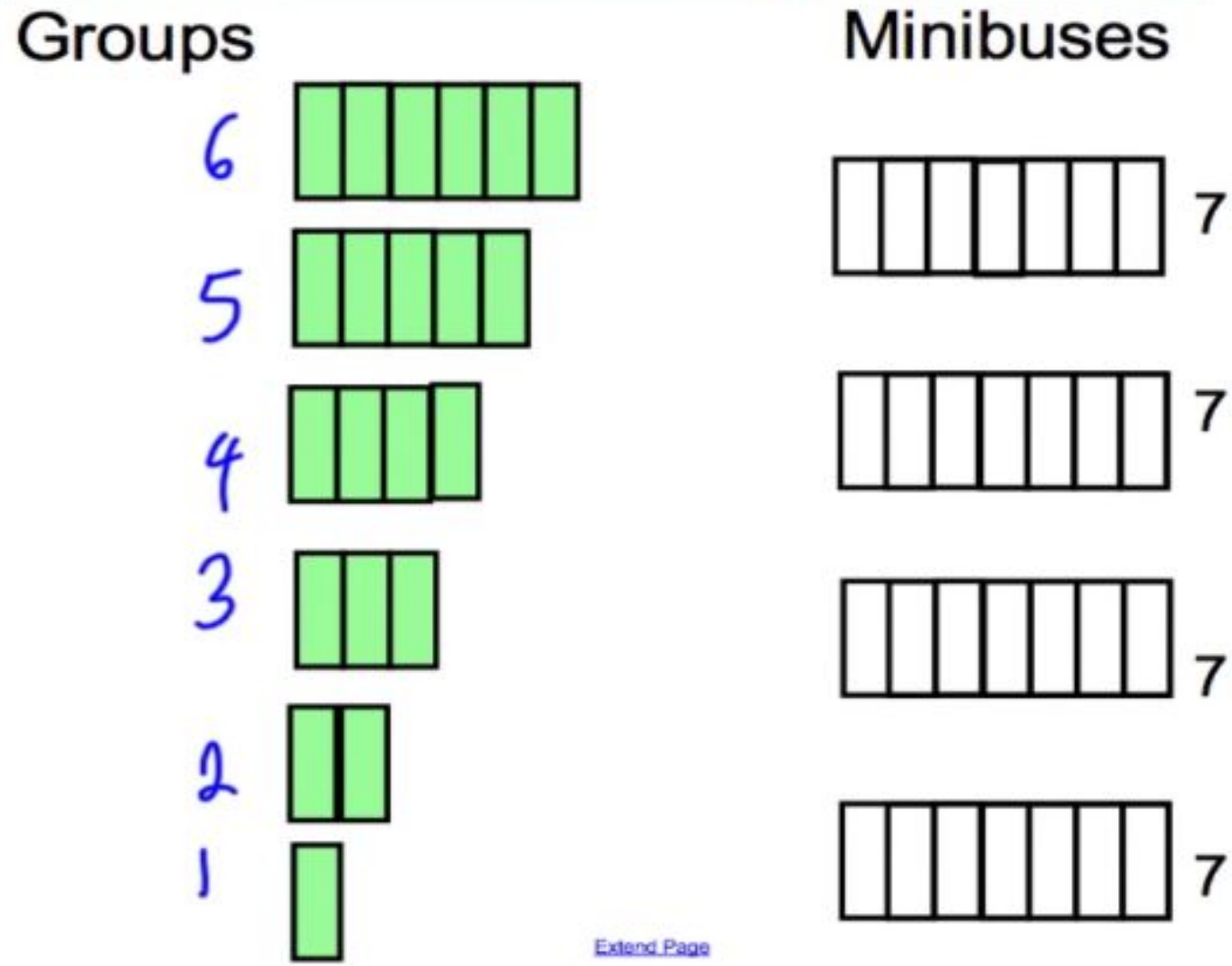
Extend Page

First-fit decreasing algorithm



Extend Page

First-fit decreasing algorithm



First-fit decreasing algorithm

Groups

6

5

4

3

2

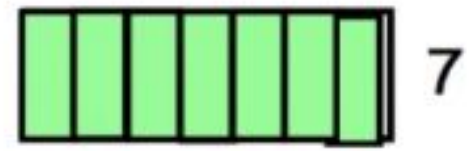
1

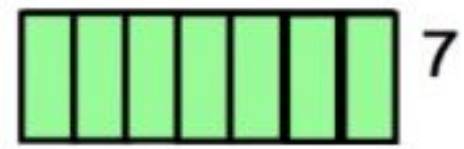
3 mini-buses

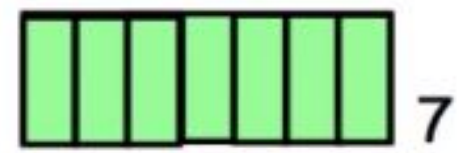
No space

Optimal.

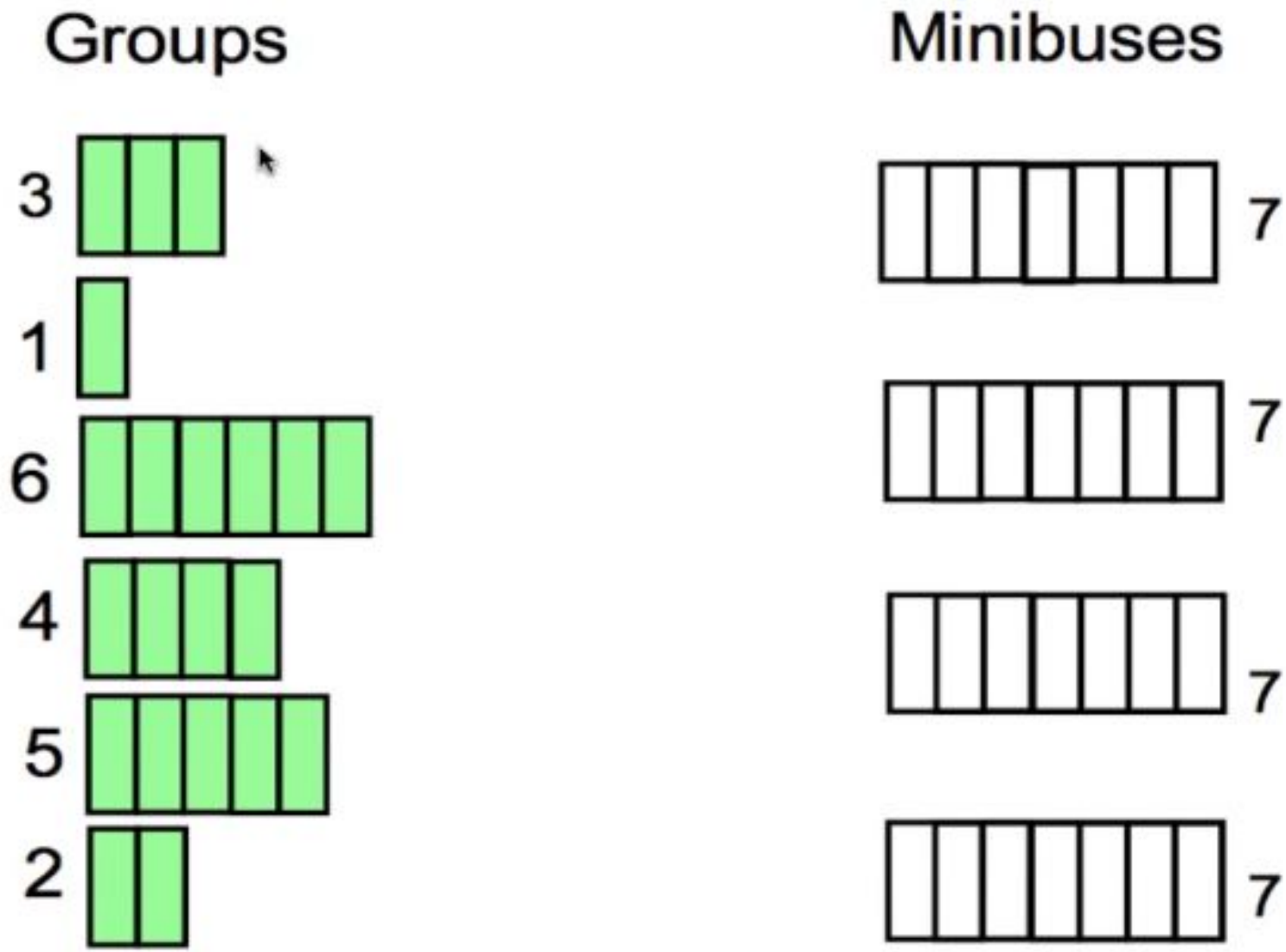
Minibuses





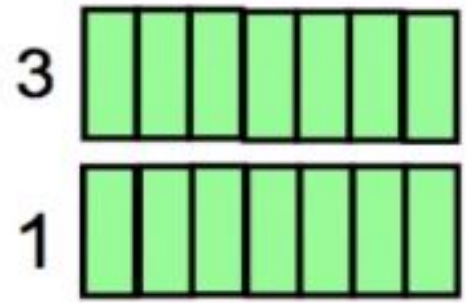


Full-bin packing

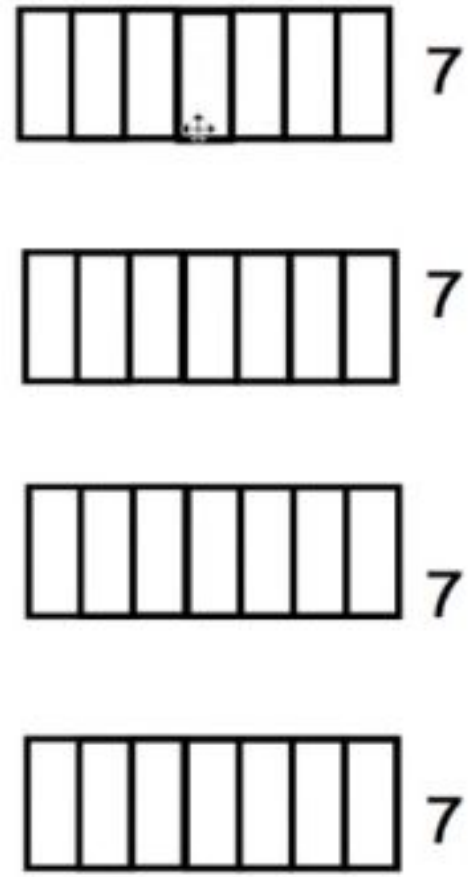


Full-bin packing

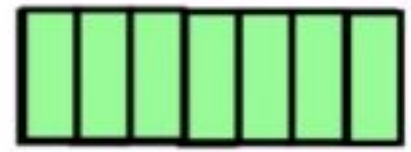
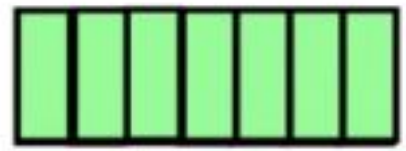
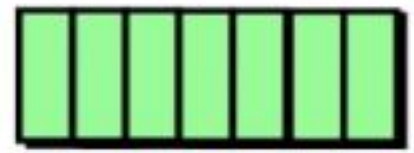
Groups



Minibuses



Full-bin packing

Groups		Minibuses	
3	3 minibuses		7
1	No space		7
6	Minimal.		
4			7
5			
2			7

Lower Bound

Example 1:

A plumber is using lengths of pipe 12 feet long and wishes to cut these lengths

Length (feet)	Number
2	2
3	4
4	3
6	1
7	2

$2 \times 2 = 4$
$4 \times 3 = 12$
$3 \times 4 = 12$
$1 \times 6 = 6$
$2 \times 7 = 14$
48

$48 \div 12 = 4$ pipes

Lower Pipes

First-fit algorithm

Example 2:

A plumber is using lengths of pipe 12 feet long and wishes to cut these lengths.

Length (feet)	Number
2	2
3	4
4	3
6	1
7	2

2, 2, 3, 3, 3, 4, 4, 4, 6, 7, 7

12: 2 2 3 3 — w 2
12: 3 3 4 — w 2
12: 4 4 — w 4
12: 6 — w 6
12: 7 — w 5
12: 7 — w 5

6 pipes

24 ft wasted

First-fit decreasing algorithm

Example 3:

A plumber is using lengths of pipe 12 feet long and wishes to cut these lengths.

Length (feet)	Number
2	2
3	4
4	3
6	1
7	2

2, 2, 3, 3, 3, 3, 4, 4, 4, 6, 7, 7

12: 7, 4 W1

12: 7, 4 W1

12: 6, 4, 2

12: 3, 3, 3, 3 W10

12: 2

5 pipes

12 ft waste

Extend Page

Full-bin packing

Example 4:

A plumber is using lengths of pipe 12 feet long and wishes to cut these lengths.

Length (feet)	Number
2	2
3	4
4	3
6	1
7	2

2, 2, 3, 3, 3, 3, 4, 4, 4, 6, 7, 7

7, 3, 2

7, 3, 2

4, 4, 4

3, 3, 6

4 pipes

0 waste

First-fit algorithm

- Quick and easy to perform
- Does not usually lead to an optimal solution

First-fit decreasing algorithm

- Quick and easy to perform
- Usually better solution than first-fit
- Do not always get an optimal solution

Full-bin packing

- Usually gets a good solution
- Can be difficult to perform, if numbers awkward

Textbooks & Web References

- Text Book (Chapter 16 and 35)
- Reference book iii (Chapter 17)
- www.geeksforgeeks.org

Thank you

&

Any question?